FIG.1

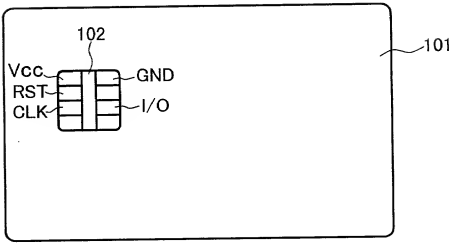

102
101
Vcc
RST
CLK
GND
I/O

FIG.2

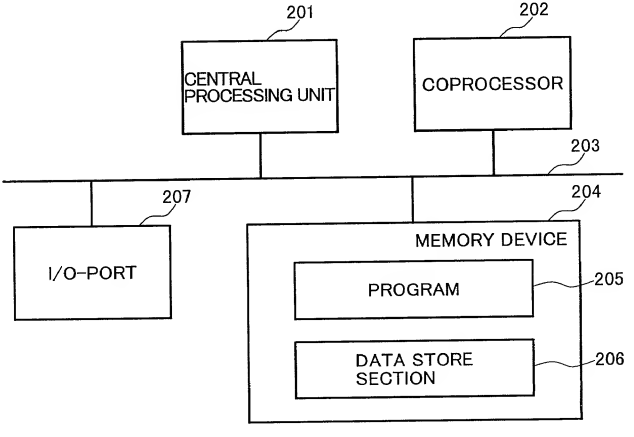

201
CENTRAL PROCESSING UNIT
202
COPROCESSOR
203
207
I/O-PORT
204
MEMORY DEVICE
PROGRAM
205
DATA STORE SECTION
206

FIG.3

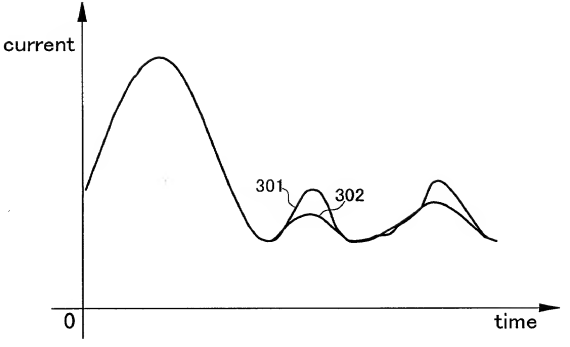


FIG.4

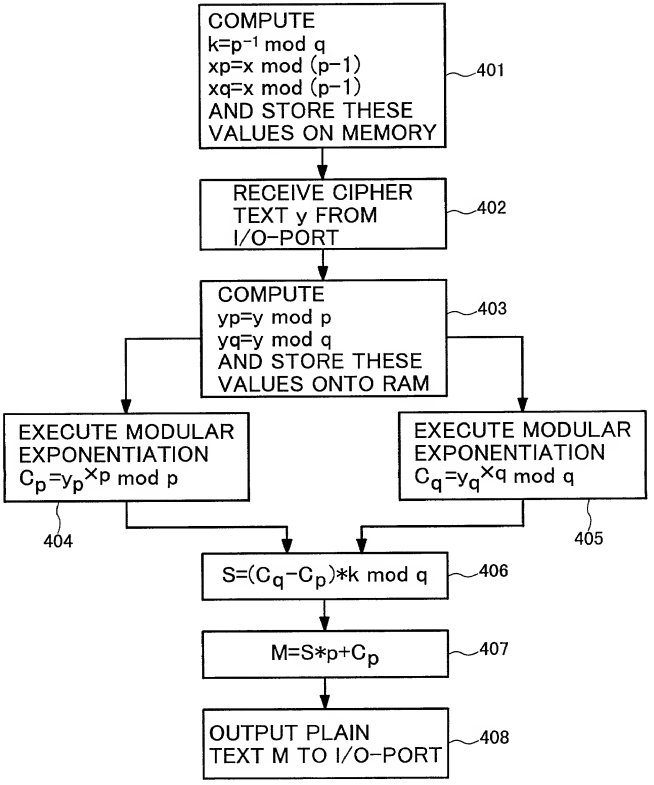

COMPUTE
k=p^-1 mod q
xp=x mod (p-1)
xq=x mod (p-1)
AND STORE THESE
VALUES ON MEMORY
401
RECEIVE CIPHER
TEXT y FROM
I/O-PORT
402
COMPUTE
yp=y mod p
yq=y mod q
AND STORE THESE
VALUES ONTO RAM
403
EXECUTE MODULAR
EXPONENTIATION
Cp=yp^Xp mod p
404
EXECUTE MODULAR
EXPONENTIATION
Cq=yq^Xq mod q
405
S=(Cq-Cp)*k mod q
406
M=S*p+Cp
407
OUTPUT PLAIN
TEXT M TO I/O-PORT
408

FIG.5

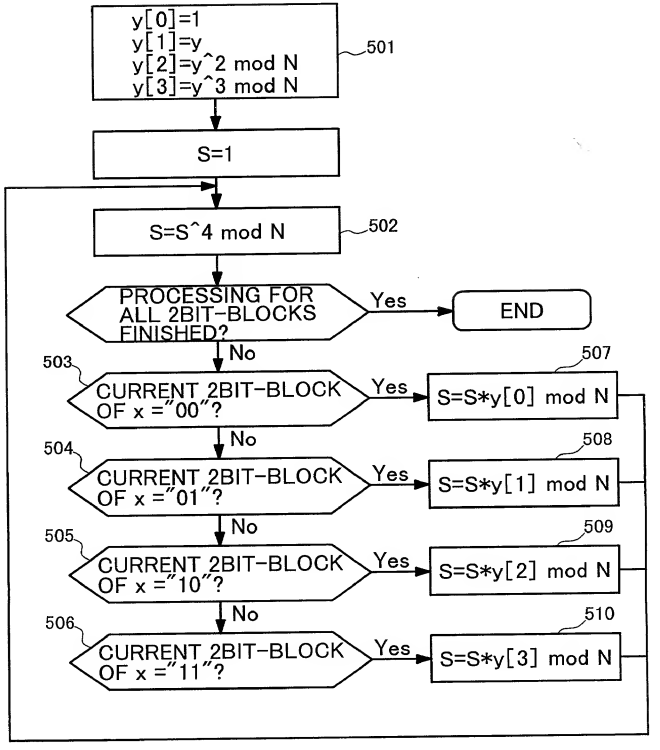

y[0]=1
y[1]=y
y[2]=y^2 mod N
y[3]=y^3 mod N
501
S=1
S=S^4 mod N
502
PROCESSING FOR ALL 2BIT-BLOCKS FINISHED?
Yes
END
No
503
CURRENT 2BIT-BLOCK OF x ="00"?
Yes
507
S=S*y[0] mod N
No
504
CURRENT 2BIT-BLOCK OF x ="01"?
Yes
508
S=S*y[1] mod N
No
505
CURRENT 2BIT-BLOCK OF x ="10"?
Yes
509
S=S*y[2] mod N
No
506
CURRENT 2BIT-BLOCK OF x ="11"?
Yes
510
S=S*y[3] mod N

FIG.6

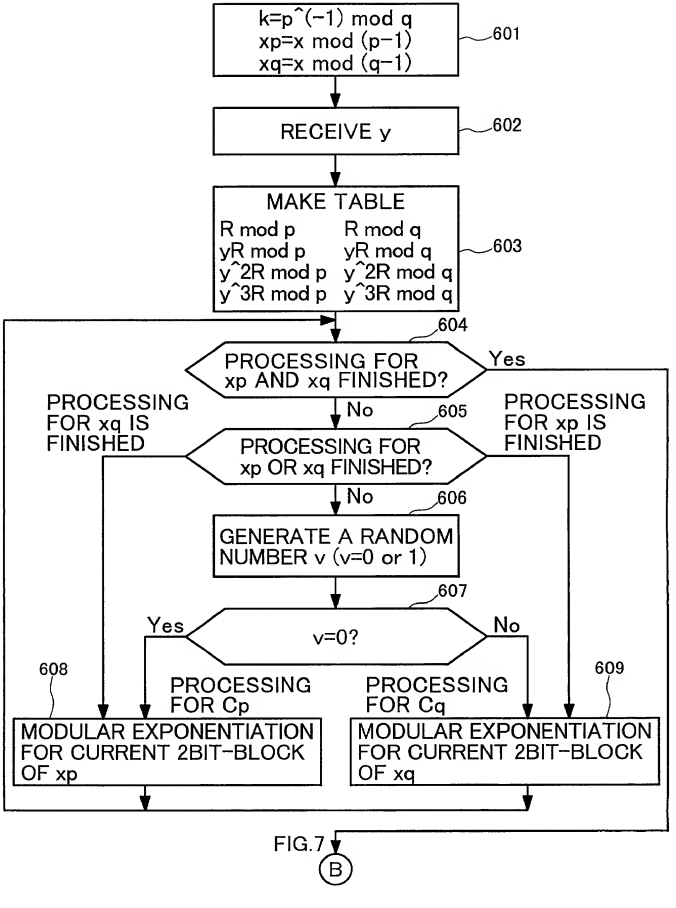

k=p^(-1) mod q
xp=x mod (p-1)
xq=x mod (q-1)
601
RECEIVE y
602
MAKE TABLE
R mod p
R mod q
yR mod p
yR mod q
y^2R mod p
y^2R mod q
y^3R mod p
y^3R mod q
603
604
PROCESSING FOR xp AND xq FINISHED?
Yes
No
PROCESSING FOR xq IS FINISHED
PROCESSING FOR xp IS FINISHED
605
PROCESSING FOR xp OR xq FINISHED?
No
606
GENERATE A RANDOM NUMBER v (v=0 or 1)
607
Yes
v=0?
No
608
609
PROCESSING FOR Cp
PROCESSING FOR Cq
MODULAR EXPONENTIATION FOR CURRENT 2BIT-BLOCK OF xp
MODULAR EXPONENTIATION FOR CURRENT 2BIT-BLOCK OF xq
FIG.7
B

FIG.7

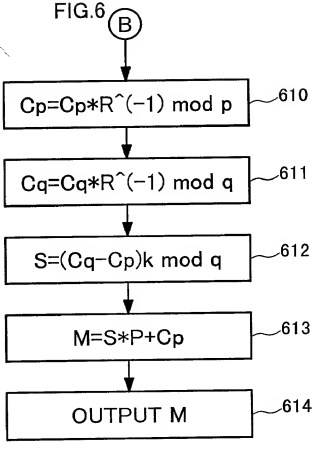


FIG.6
B
Cp=Cp*R^(-1) mod p
610
Cq=Cq*R^(-1) mod q
611
S=(Cq-Cp)k mod q
612
M=S*P+Cp
613
OUTPUT M
614

FIG.8

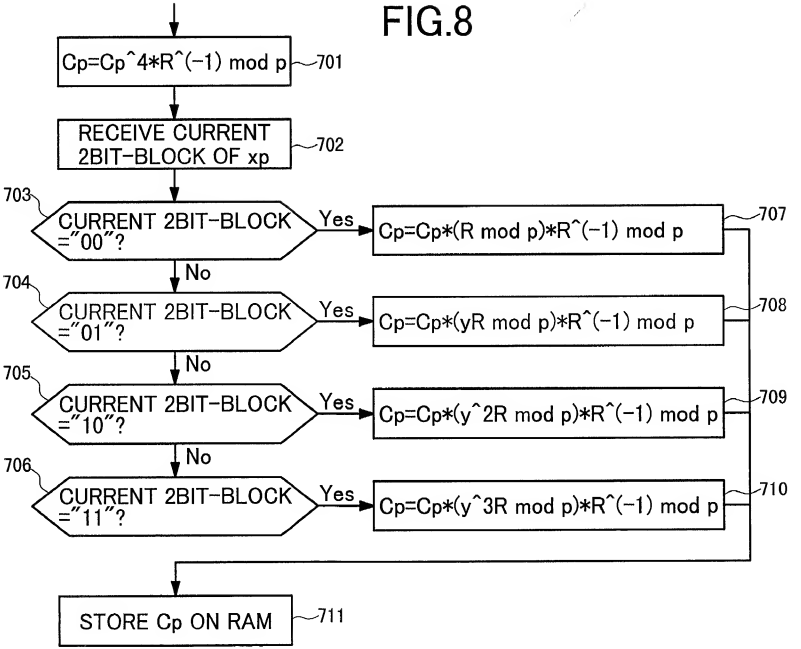

Cp=Cp^4*R^(-1) mod p
701
RECEIVE CURRENT 2BIT-BLOCK OF xp
702
703
CURRENT 2BIT-BLOCK ="00"?
Yes
Cp=Cp*(R mod p)*R^(-1) mod p
707
No
704
CURRENT 2BIT-BLOCK ="01"?
Yes
Cp=Cp*(yR mod p)*R^(-1) mod p
708
No
705
CURRENT 2BIT-BLOCK ="10"?
Yes
Cp=Cp*(y^2R mod p)*R^(-1) mod p
709
No
706
CURRENT 2BIT-BLOCK ="11"?
Yes
Cp=Cp*(y^3R mod p)*R^(-1) mod p
710
STORE Cp ON RAM
711

FIG.9

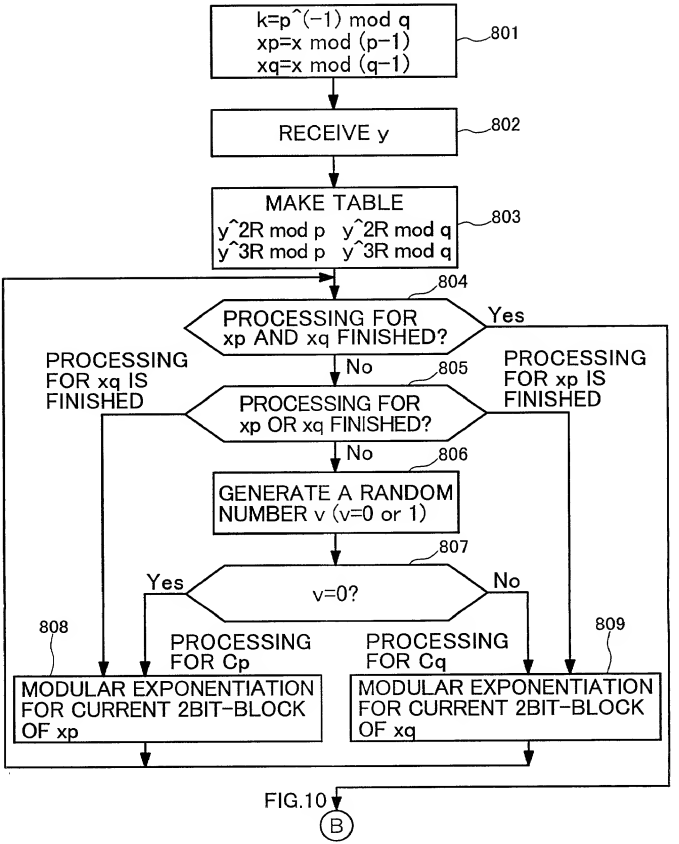

k=p^(−1) mod q
xp=x mod (p−1)
xq=x mod (q−1)
801
RECEIVE y
802
MAKE TABLE
y^2R mod p y^2R mod q
y^3R mod p y^3R mod q
803
804
PROCESSING FOR xp AND xq FINISHED?
Yes
No
PROCESSING FOR xq IS FINISHED
PROCESSING FOR xp IS FINISHED
805
PROCESSING FOR xp OR xq FINISHED?
No
806
GENERATE A RANDOM NUMBER v (v=0 or 1)
807
Yes
v=0?
No
808
PROCESSING FOR Cp
PROCESSING FOR Cq
809
MODULAR EXPONENTIATION FOR CURRENT 2BIT-BLOCK OF xp
MODULAR EXPONENTIATION FOR CURRENT 2BIT-BLOCK OF xq

FIG.10

B

FIG.10

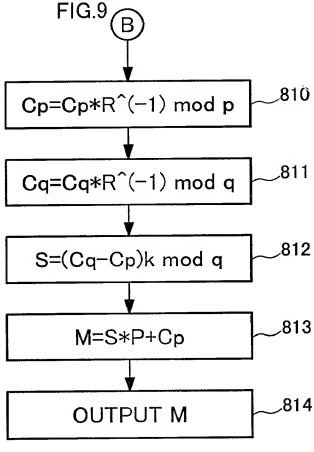


FIG.9
B
Cp=Cp*R^(-1) mod p
810
Cq=Cq*R^(-1) mod q
811
S=(Cq-Cp)k mod q
812
M=S*P+Cp
813
OUTPUT M
814

FIG.11

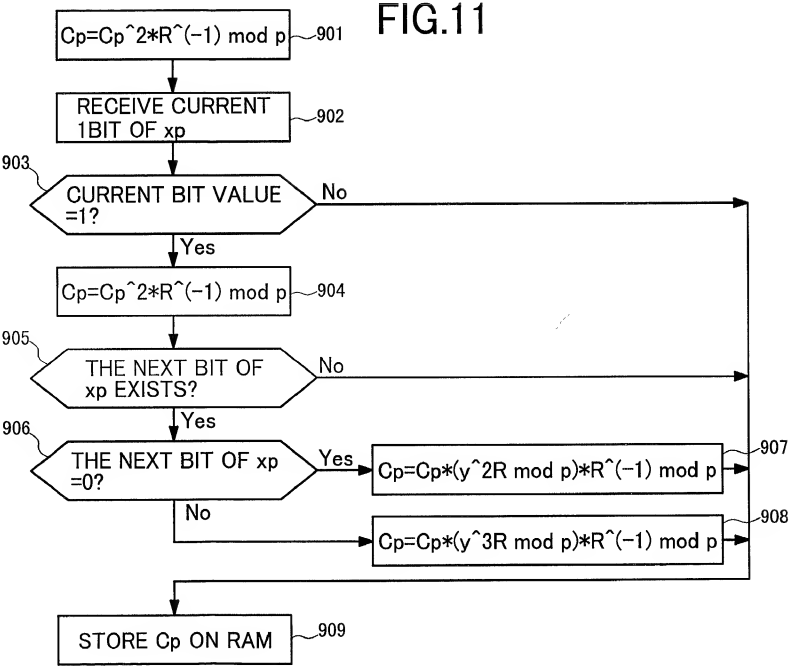

Cp=Cp^2*R^(-1) mod p
901
RECEIVE CURRENT 1BIT OF xp
902
903
CURRENT BIT VALUE =1?
No
Yes
Cp=Cp^2*R^(-1) mod p
904
905
THE NEXT BIT OF xp EXISTS?
No
Yes
906
THE NEXT BIT OF xp =0?
Yes
Cp=Cp*(y^2R mod p)*R^(-1) mod p
907
No
Cp=Cp*(y^3R mod p)*R^(-1) mod p
908
STORE Cp ON RAM
909

FIG.12

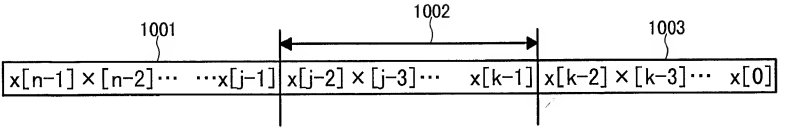


FIG.13

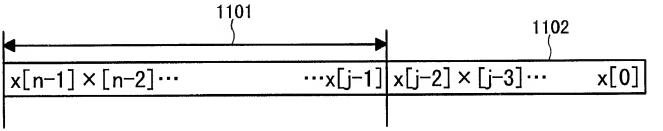


FIG.14

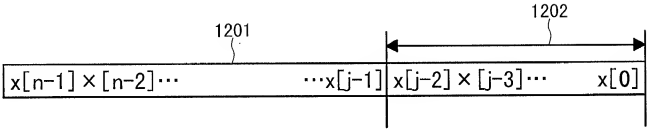
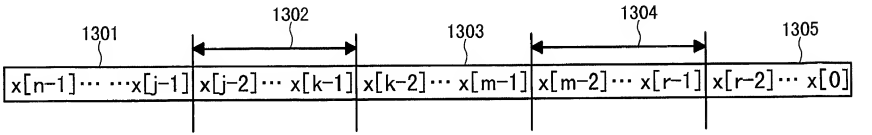


FIG.15



TD6280 5860A660

FIG.16A

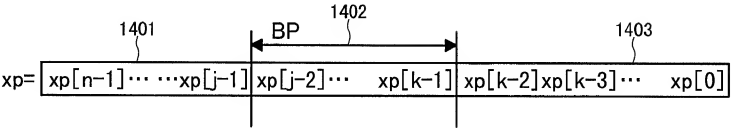


FIG.16B

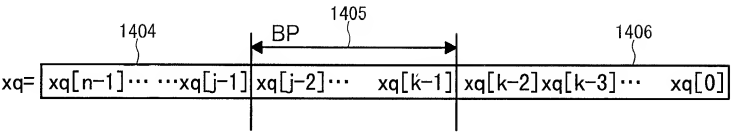


FIG.17

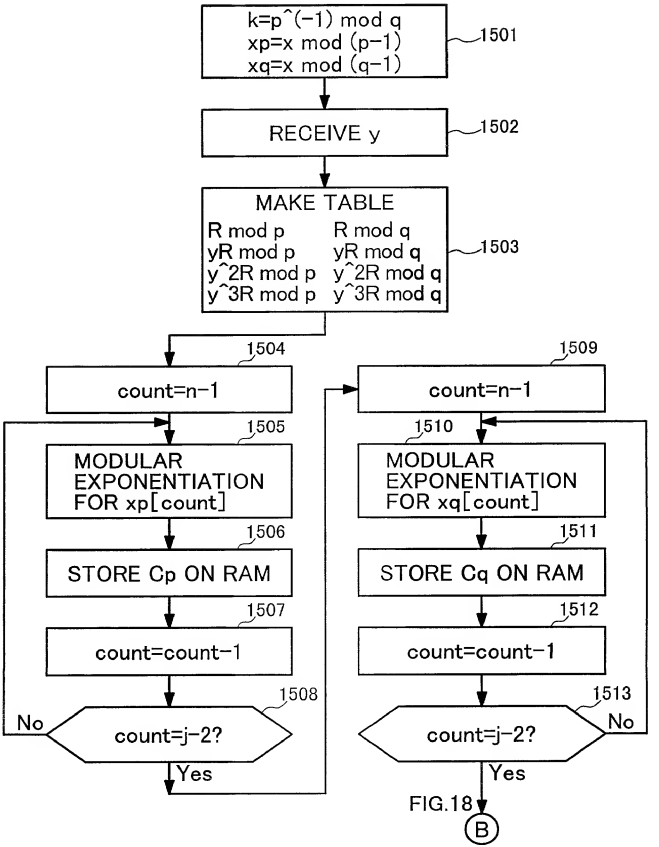

k=p^(-1) mod q
xp=x mod (p-1)
xq=x mod (q-1)
1501
RECEIVE y
1502
MAKE TABLE
R mod p
yR mod p
y^2R mod p
y^3R mod p
R mod q
yR mod q
y^2R mod q
y^3R mod q
1503
1504
count=n-1
1505
MODULAR EXPONENTIATION FOR xp[count]
1506
STORE Cp ON RAM
1507
count=count-1
1508
No
count=j-2?
Yes
1509
count=n-1
1510
MODULAR EXPONENTIATION FOR xq[count]
1511
STORE Cq ON RAM
1512
count=count-1
1513
count=j-2?
No
Yes
FIG.18
B

FIG.18

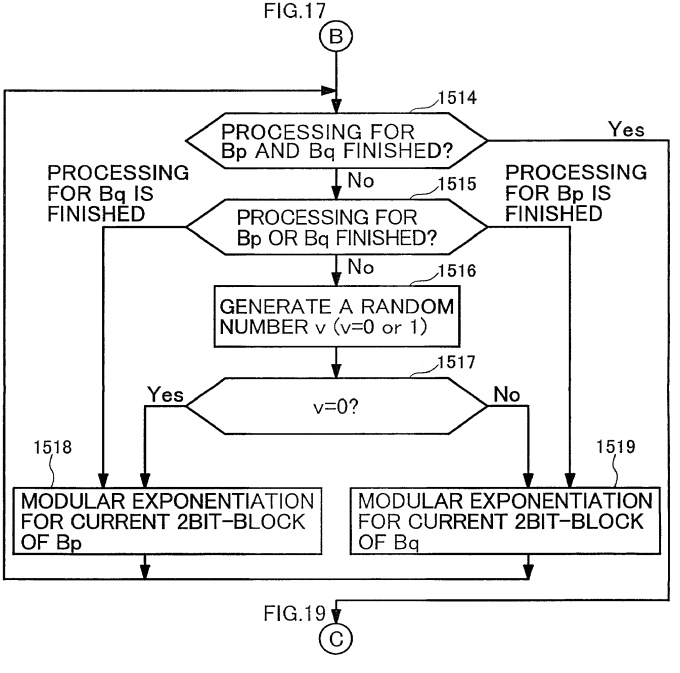


FIG.17
B
1514
PROCESSING FOR Bp AND Bq FINISHED?
Yes
PROCESSING FOR Bq IS FINISHED
No
1515
PROCESSING FOR Bp IS FINISHED
PROCESSING FOR Bp OR Bq FINISHED?
No
1516
GENERATE A RANDOM NUMBER v (v=0 or 1)
1517
Yes
v=0?
No
1518
1519
MODULAR EXPONENTIATION FOR CURRENT 2BIT-BLOCK OF Bp
MODULAR EXPONENTIATION FOR CURRENT 2BIT-BLOCK OF Bq
FIG.19
C

FIG.18 (C)

FIG.19

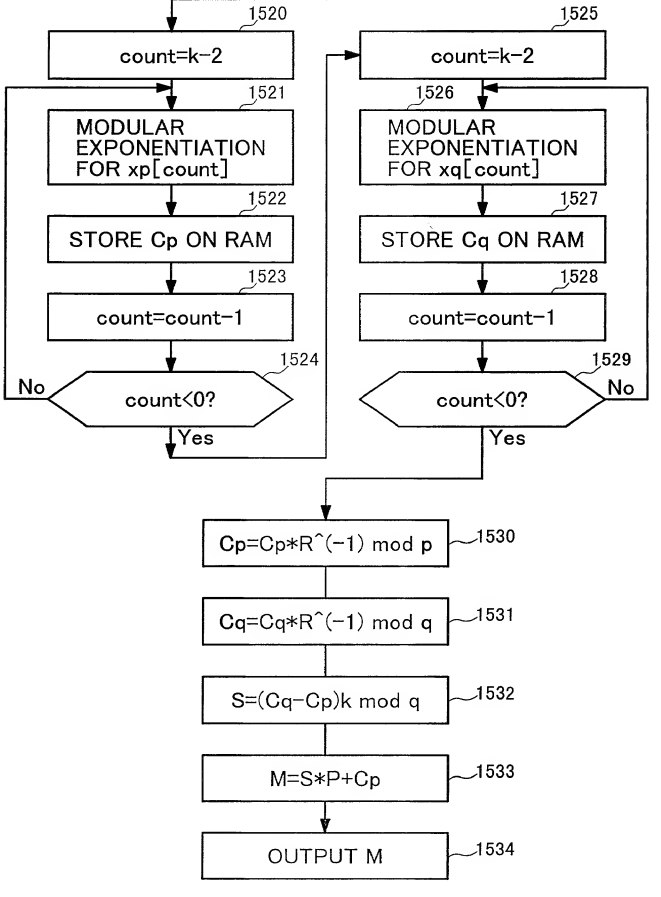


FIG.20

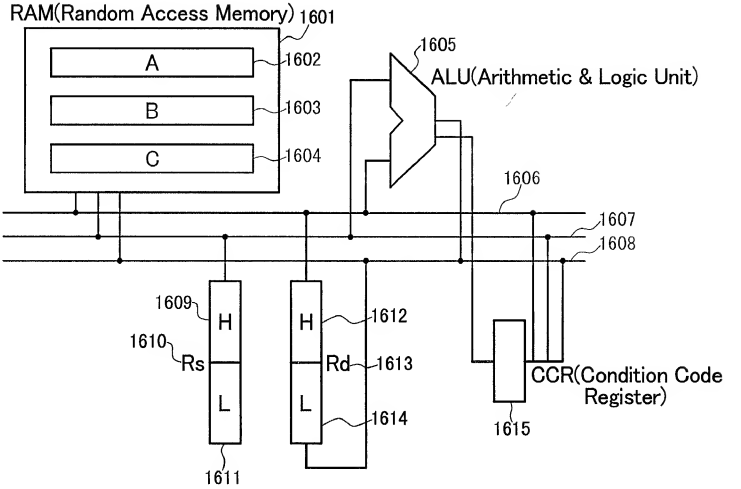

RAM(Random Access Memory)
1601
A
1602
B
1603
C
1604
1605
ALU(Arithmetic & Logic Unit)
1606
1607
1608
1609
H
1610
Rs
L
1611
H
1612
Rd
1613
L
1614
CCR(Condition Code Register)
1615

FIG.21

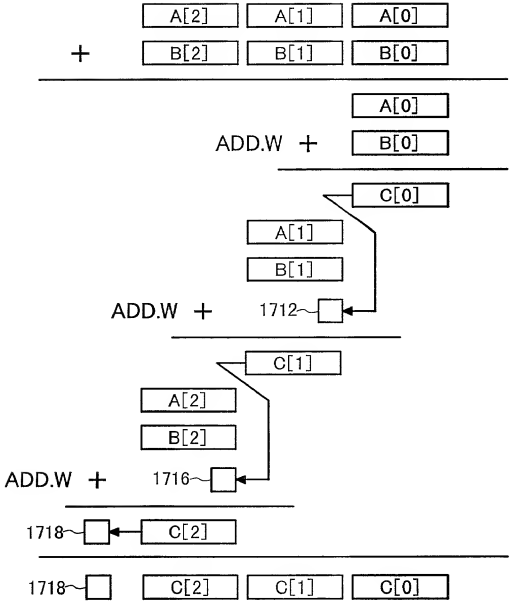

A[2]
A[1]
A[0]
+
B[2]
B[1]
B[0]
A[0]
ADD.W +
B[0]
C[0]
A[1]
B[1]
ADD.W +
1712
C[1]
A[2]
B[2]
ADD.W +
1716
1718
C[2]
1718
C[2]
C[1]
C[0]

FIG.22

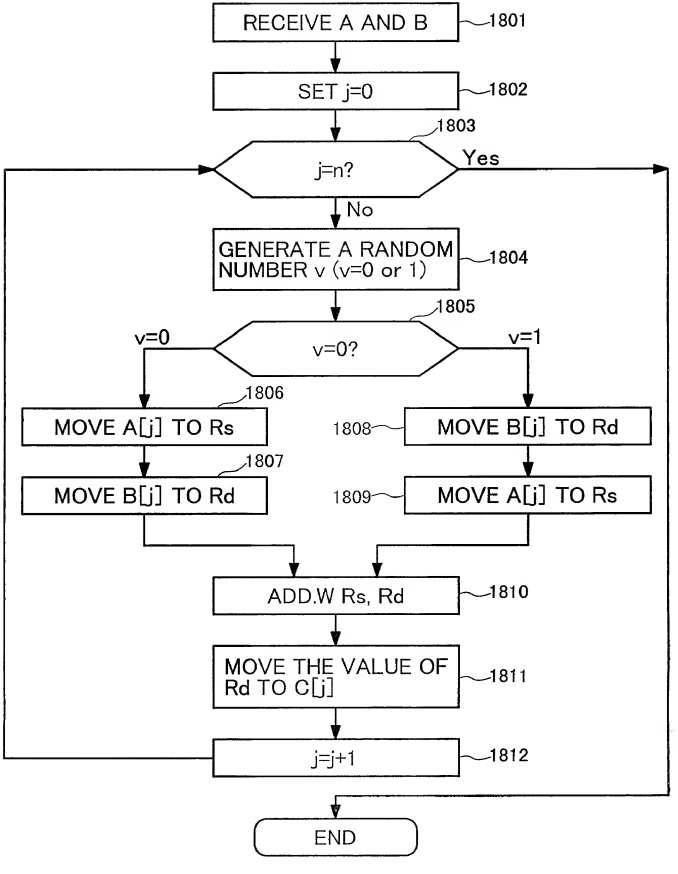

RECEIVE A AND B
1801
SET j=0
1802
1803
j=n?
Yes
No
GENERATE A RANDOM NUMBER v (v=0 or 1)
1804
1805
v=0
v=0?
v=1
1806
MOVE A[j] TO Rs
1808
MOVE B[j] TO Rd
1807
MOVE B[j] TO Rd
1809
MOVE A[j] TO Rs
ADD.W Rs, Rd
1810
MOVE THE VALUE OF Rd TO C[j]
1811
j=j+1
1812
END

FIG.23

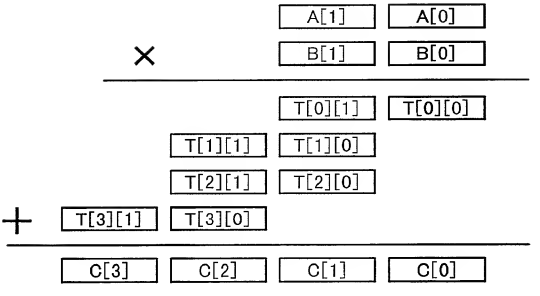


FIG.24

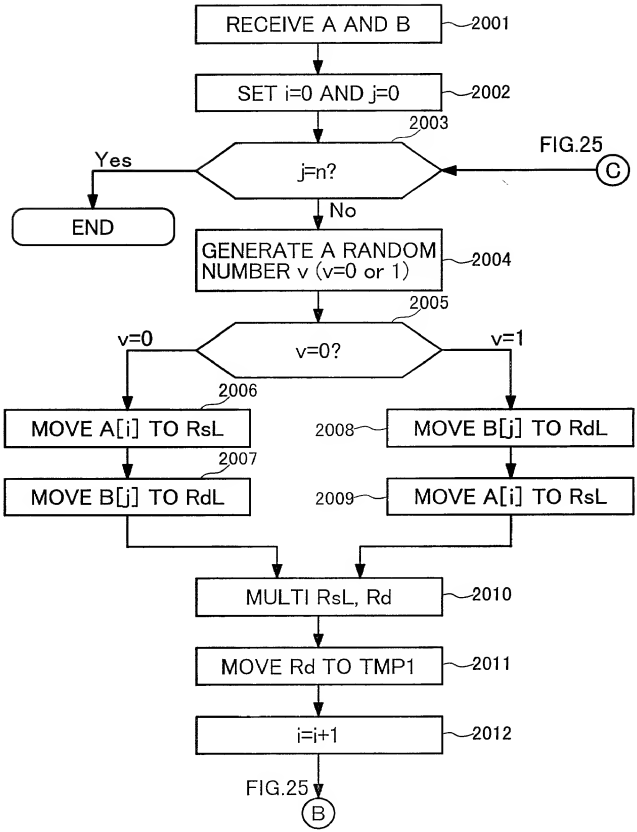

RECEIVE A AND B
2001
SET i=0 AND j=0
2002
2003
FIG.25
C
Yes
j=n?
END
No
GENERATE A RANDOM NUMBER v (v=0 or 1)
2004
2005
v=0
v=0?
v=1
2006
MOVE A[i] TO RsL
2008
MOVE B[j] TO RdL
2007
MOVE B[j] TO RdL
2009
MOVE A[i] TO RsL
MULTI RsL, Rd
2010
MOVE Rd TO TMP1
2011
i=i+1
2012
FIG.25
B

FIG.24

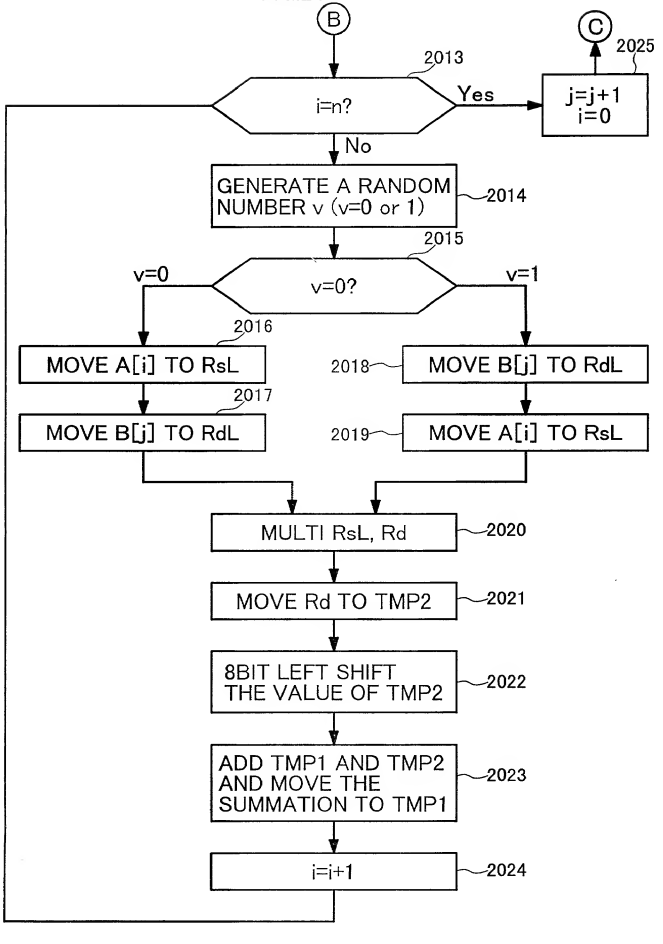

B
C
2013
i=n?
Yes
2025
j=j+1
i=0
No
GENERATE A RANDOM NUMBER v (v=0 or 1)
2014
2015
v=0
v=0?
v=1
2016
MOVE A[i] TO RsL
2018
MOVE B[j] TO RdL
2017
MOVE B[j] TO RdL
2019
MOVE A[i] TO RsL
MULTI RsL, Rd
2020
MOVE Rd TO TMP2
2021
8BIT LEFT SHIFT THE VALUE OF TMP2
2022
ADD TMP1 AND TMP2 AND MOVE THE SUMMATION TO TMP1
2023
i=i+1
2024

FIG.25

FIG.26

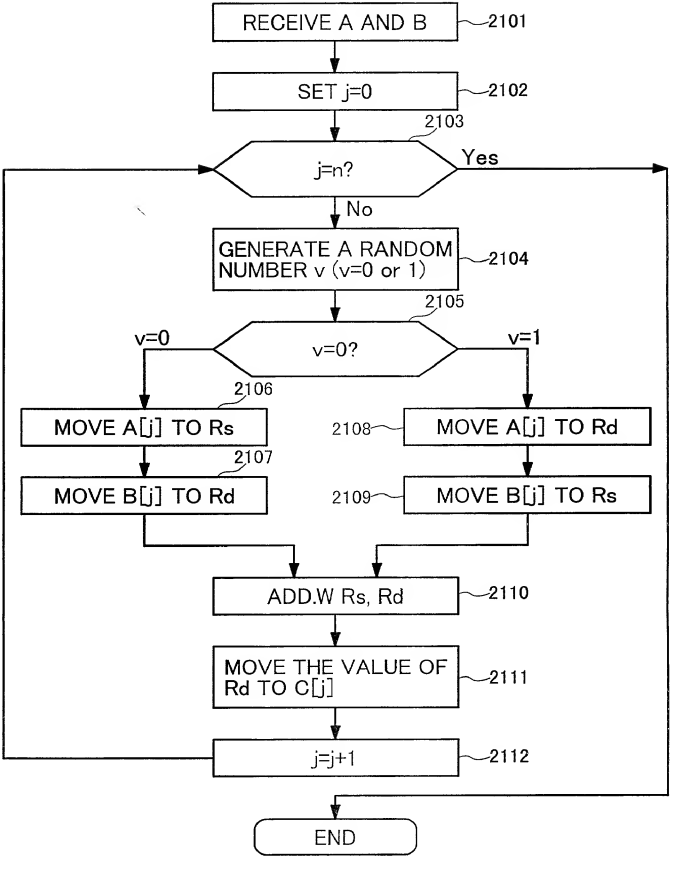

RECEIVE A AND B
2101
SET j=0
2102
2103
j=n?
Yes
No
GENERATE A RANDOM NUMBER v (v=0 or 1)
2104
2105
v=0?
v=0
v=1
2106
MOVE A[j] TO Rs
2107
MOVE B[j] TO Rd
2108
MOVE A[j] TO Rd
2109
MOVE B[j] TO Rs
ADD.W Rs, Rd
2110
MOVE THE VALUE OF Rd TO C[j]
2111
j=j+1
2112
END

FIG.27

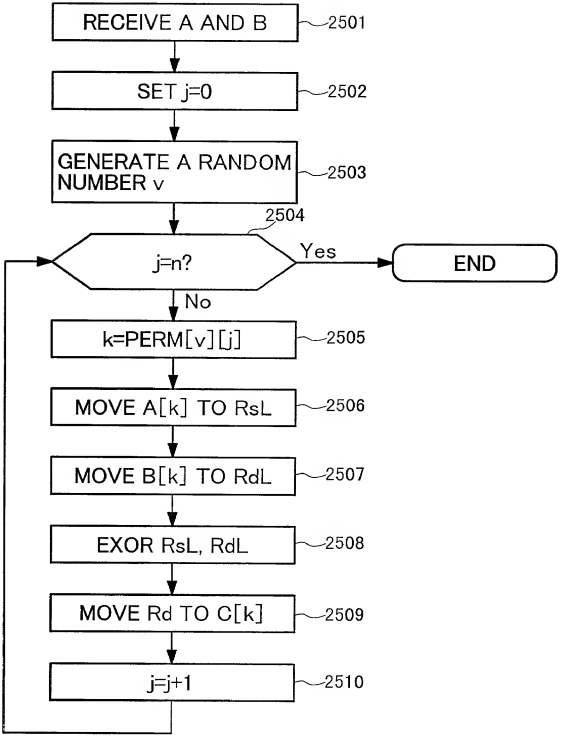

RECEIVE A AND B
2501
SET j=0
2502
GENERATE A RANDOM NUMBER v
2503
2504
j=n?
Yes
END
No
k=PERM[v][j]
2505
MOVE A[k] TO RsL
2506
MOVE B[k] TO RdL
2507
EXOR RsL, RdL
2508
MOVE Rd TO C[k]
2509
j=j+1
2510

FIG.28

A	B	A EXOR B
1	1	0
1	0	1
0	1	1
0	0	0

FIG.29

A	B	A AND B
1	1	1
1	0	0
0	1	0
0	0	0

FIG.30

A	B	A OR B
1	1	1
1	0	1
0	1	1
0	0	0